UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/762,814	01/22/2004	David Bau	ORACL-01388US1	6120

80548 7590 04/28/2009

Fliesler Meyer LLP
650 California Street
14th Floor
San Francisco, CA 94108

EXAMINER	
CHEN, QING	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
04/28/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/762,814	BAU, DAVID	
	Examiner	Art Unit	
	Qing Chen	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *09 February 2009*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-64* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-64* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is in response to the amendment filed on February 9, 2009.

2. **Claims 1-64** are pending.

3. **Claims 1-33, 35-37, 40-54, and 56-64** have been amended.

4. **Claims 65 and 66** have been canceled.

5. The objection to the title is maintained in view of Applicant's amendments to the title and further explained hereinafter.

6. The objection to the abstract is withdrawn in view of Applicant's amendments to the abstract.

7. The objections to the specification are withdrawn in view of Applicant's amendments to the specification.

8. The objections to Claims 2-22, 24, 28, 37-39, 41, 45, 49, and 59-60 are withdrawn in view of Applicant's amendments to the claims. However, Applicant's amendments to the claims fail to address the objections to Claims 40 and 58 due to typographical errors. Accordingly, these objections are maintained and further explained hereinafter.

9. The 35 U.S.C. § 112, first paragraph, rejections of Claims 1-65 are withdrawn in view of Applicant's amendments to the claims or cancellation of the claims.

10. The 35 U.S.C. § 112, second paragraph, rejections of Claims 1-25, 27-39, 41-63, and 65 are withdrawn in view of Applicant's amendments to the claims or cancellation of the claims. However, Applicant's amendments to the claims fail to fully address the rejections to Claims 26, 40, and 64 due to insufficient antecedent bases and the use of trademarks. Accordingly, these rejections are maintained and further explained hereinafter.

11. The 35 U.S.C. § 101 rejections of Claims 40-43 and 65 are withdrawn in view of Applicant's amendments to the claims or cancellation of the claims. However, the 35 U.S.C. § 101 rejections of Claims 1-22 are maintained in view of Applicant's amendments to the claims and further explained hereinafter.

12. It is noted that Claim 55 contains proposed amendments. However, the claim still bears the "Original" status identifier.

13. It is noted that Claims 19, 41, 44, and 61 contain either erroneous or missing amendments that have been made relative to the immediate prior version of the claims.

Response to Amendment

Specification

14. The title of the invention is objected to because the trademark or trade name JAVA should be accompanied with an appropriate designation symbol, *e.g.*, ™ or ®.

Appropriate correction is required.

Claim Objections

15. **Claims 40 and 58** are objected to because of the following informalities:

- **Claim 40** contains a typographical error: "[I]s capable of […]" should read -- and is capable of […] --.
- **Claim 58** contains the following typographical errors:
  - "[A] XML schema" should read -- an XML schema --.
  - "[T]he source and target type" should read -- the source and target types --.

Appropriate correction is required.

Claim Rejections - 35 USC § 112

16. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

17. **Claims 15-17** are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

Claim 15 recites the limitation of accessing the XML data from within the object-oriented programming language without mapping the XML data to an object-oriented programming language object. The subject matter is not properly described in the application as filed, since the specification only discloses that a registry can be used that allows a single Java type to map to any number of different XML types *(see page 9, paragraph [0043])* and a map is being referenced between the XML and the Java types *(see page 13, paragraph [0056])*. The specification lacks disclosure on without mapping an XML type to a Java type. Because the specification does not adequately support the claimed subject matter, it would not reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

Claims 16 and 17 depend on Claim 15 and, therefore, suffer the same deficiency as Claim 15.

18. **Claims 15-17** are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

Claim 15 recites the limitation of accessing the XML data from within the object-oriented programming language without mapping the XML data to an object-oriented programming language object. Nowhere in the Applicant's specification does the Applicant

provide an enabling disclosure on accessing XML data from within Java without mapping the XML data to a Java object. Applicant's specification only describes mapping an XML type to a Java type *(see, e.g., page 9, paragraph [0043] and page 13, paragraph [0056])*. Thus, the Applicant's specification does not enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention. In the interest of compact prosecution, the Examiner subsequently interprets this limitation as reading "accessing the XML data from within the object-oriented programming language by mapping the XML data to an object-oriented programming language object" for the purpose of further examination.

Claims 16 and 17 depend on Claim 15 and, therefore, suffer the same deficiency as Claim 15.

19. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

20. **Claims 26, 40, and 64** are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 26 contains the trademark or trade name JAVA. When a trademark or trade name is used in a claim as a limitation to identify or describe a particular material or product, the claim does not comply with the requirements of the 35 U.S.C. 112, second paragraph. *Ex parte Simpson*, 218 USPQ 1020 (Bd. App. 1982). The claim scope is uncertain since the trademark or

trade name cannot be used properly to identify any particular material or product. A trademark or trade name is used to identify a source of goods, and not the goods themselves. Thus, the use of a trademark or trade name in a claim to identify or describe a material or product (in the present case, a specific programming language) would not only render a claim indefinite, but would also constitute an improper use of the trademark or trade name.

Claim 40 recites the limitation "the XML schema." There is insufficient antecedent basis for this limitation in the claim. In the interest of compact prosecution, the Examiner subsequently interprets this limitation as reading "an XML schema" for the purpose of further examination.

Claim 64 recites the limitation "the XML type." There is insufficient antecedent basis for this limitation in the claim. In the interest of compact prosecution, the Examiner subsequently interprets this limitation as reading "the object-oriented programming language type" for the purpose of further examination.

Claim Rejections - 35 USC § 101

21. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

22. **Claims 1-22** are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claims 1-22 are directed to computer-implemented systems. However, the recited components of the computer-implemented systems appear to lack the necessary physical components (hardware) to constitute a machine or manufacture under § 101. Although the claims recite the systems as being implemented on a computer or a compiler as being run on one or more processors, however, such recitation can be construed to be an intended use of the computer and the processor rather than physical components of the systems. Therefore, these claim limitations can be reasonably interpreted as computer program modules—software *per se*. The claims are directed to functional descriptive material *per se*, and hence non-statutory.

The claims constitute computer programs representing computer listings *per se*. Such descriptions or expressions of the programs are not physical "things." They are neither computer components nor statutory processes, as they are not "acts" being performed. Such claimed computer programs do not define any structural and functional interrelationships between the computer program and other claimed elements of a computer, which permit the computer program's functionality to be realized. In contrast, a claimed computer-readable medium encoded with a computer program is a computer element, which defines structural and functional interrelationships between the computer program and the rest of the computer, that permits the computer program's functionality to be realized, and is thus statutory. See *Lowry*, 32 F.3d at 1583-84, 32 USPQ2d at 1035.

Claim Rejections - 35 USC § 102

23. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

24. **Claims 1, 3-23, 25-44, and 46-64** are rejected under 35 U.S.C. 102(e) as being anticipated by **US 7,155,705 (hereinafter "Hershberg")**.

As per **Claim 1**, Hershberg discloses:

- an XML data *(see Figure 1A: 118; Column 7: 54-58, "The XML configuration information 116 includes the user-defined JavaDoc tag "@xml-root-element" that is an example of tag 118 ...")*;

- an XML schema which defines the XML data *(see Column 7: 54-58, "The XML configuration information 116 includes the user-defined JavaDoc tag "@xml-root-element" that is an example of tag 118, and the property "dtdFile." The property is set to a value "employee.dtd" given between the quotation marks in line 3.5.")*;

- an object-oriented programming language type which corresponds to the XML schema and provides XML-oriented data manipulation, wherein the object-oriented programming language type allows the combination of XML and object-oriented programming language type systems and is capable of accessing and manipulating the XML data from within the object-oriented programming language *(see Column 14: 47-55, "In step 340, a module is generated that marshals a data object of the class to one or more data items in the exchange format. For example, a JAVA module [object-oriented programming language type] is generated*

that marshals a data object of a JAVA class into one or more items of the XML configured by the DTD [XML-oriented data manipulation]. As a more specific example, a JAVA module is generated that exports an object of class Employee having attributes empname and empid into an XML element Employee having attribute empid and containing element LastName."); and

- a compiler capable of generating the object-oriented programming language type from the XML schema, wherein the compiler runs on one or more processors *(see Column 6: 37, "JAVA compilers ..."; Column 10: 12-16, "An example XML document with statements using the elements defined in the DTD file are given in Table 4 for two employees, John Smith and Jane Doe having employee identification numbers of 73645112 and 1, respectively.")*.

As per **Claim 3**, the rejection of **Claim 1** is incorporated; and Hershberg further discloses:

- the compiler is capable of generating the object-oriented programming language type based on a definition file *(see Column 6: 37, "JAVA compilers ..."; Column 14: 47-49, "In step 340, a module is generated that marshals a data object of the class to one or more data items in the exchange format.")*.

As per **Claim 4**, the rejection of **Claim 1** is incorporated; and Hershberg further discloses:

- the compiler is capable of compiling an object-oriented programming language project into one or more regular object-oriented programming language types *(see Column 6: 37,*

"JAVA compilers ..."; Column 14: 47-49, "In step 340, a module is generated that marshals a data object of the class to one or more data items in the exchange format.").

As per **Claim 5**, the rejection of **Claim 1** is incorporated; and Hershberg further discloses:

- the object-oriented programming language type can be a movable cursor, capable of reading anywhere within the XML data *(see Column 14: 61-65, "In step 350, a module is generated that de-marshals a data object of the class from one or more data items in the exchange format. For example, a JAVA module is generated that de-marshals a data object of a JAVA class from one or more data items of the XML configured by the DTD.").*

As per **Claim 6**, the rejection of **Claim 1** is incorporated; and Hershberg further discloses:

- the object-oriented programming language type can be an immovable value, capable of referencing a fixed part of the XML data *(see Column 14: 61-65, "In step 350, a module is generated that de-marshals a data object of the class from one or more data items in the exchange format. For example, a JAVA module is generated that de-marshals a data object of a JAVA class from one or more data items of the XML configured by the DTD.").*

As per **Claim 7**, the rejection of **Claim 1** is incorporated; and Hershberg further discloses:

- the object-oriented programming language type can be shared among multiple object-oriented programming language components *(see Column 14: 47-51, "In step 340, a module is generated that marshals a data object of the class to one or more data items in the exchange format. For example, a JAVA module is generated that marshals a data object of a JAVA class into one or more items of the XML configured by the DTD.")*.

As per **Claim 8**, the rejection of **Claim 1** is incorporated; and Hershberg further discloses:

- the object-oriented programming language type is capable of updating the XML data within the object-oriented programming language *(see Column 14: 51-55, "As a more specific example, a JAVA module is generated that exports an object of class Employee having attributes empname and empid into an XML element Employee having attribute empid and containing element LastName.")*.

As per **Claim 9**, the rejection of **Claim 1** is incorporated; and Hershberg further discloses:

- the object-oriented programming language type is capable of accessing and updating object-oriented programming language data using object-oriented programming language type methods *(see Column 14: 51-55, "As a more specific example, a JAVA module is generated that exports an object of class Employee having attributes empname and empid into an XML element Employee having attribute empid and containing element LastName.")*.

As per **Claim 10**, the rejection of **Claim 1** is incorporated; and Hershberg further discloses:

- the object-oriented programming language type is capable of accessing and updating a database *(see Column 1: 54-62, "Data exchange also occurs between independently developed applications running on different families of processors and a database server. Such data exchange is often accomplished with a published (open) database query language. For example, the Structured Query Language (SQL) is used for exchanging data with database servers of several database systems. As used herein, a data exchange format includes either a markup language or a database query language or both.").*

As per **Claim 11**, the rejection of **Claim 1** is incorporated; and Hershberg further discloses:

- the object-oriented programming language type is capable of a number of XML data operations, which include: querying XML data, transforming between XML types, and iterating over XML data document *(see Column 14: 61-65, "In step 350, a module is generated that de-marshals a data object of the class from one or more data items in the exchange format. For example, a JAVA module is generated that de-marshals a data object of a JAVA class from one or more data items of the XML configured by the DTD.").*

As per **Claim 12**, the rejection of **Claim 1** is incorporated; and Hershberg further discloses:

- an XML schema capable of defining the legal types of the XML data, which include constraints on data types and ranges of the XML data; and constraints on the data types and ranges of the object-oriented programming language type *(see Column 9: 35-42)*.

As per **Claim 13**, the rejection of **Claim 12** is incorporated; and Hershberg further discloses:

- the compiler is capable of generating constraints on the object-oriented programming language type from the XML schema on legal types of the XML data *(see Column 6: 37, "JAVA compilers ..."; Column 10: 12-16, "An example XML document with statements using the elements defined in the DTD file are given in Table 4 for two employees, John Smith and Jane Doe having employee identification numbers of 73645112 and 1, respectively.")*.

As per **Claim 14**, the rejection of **Claim 12** is incorporated; and Hershberg further discloses:

- the constraints on the object-oriented programming language type are capable of validating the object-oriented programming language type *(see Column 9: 44-55, "The XML DTD statement in line 1 of Table 3 indicates that the root element, the first element in the DTD file, is called Employee and that the Employee element must include the sub-element LastName only once.")*.

As per **Claim 15**, Hershberg discloses:

- an XML data *(see Figure 1A: 118; Column 7: 54-58, "The XML configuration information 116 includes the user-defined JavaDoc tag "@xml-root-element" that is an example of tag 118 ...")*;

- an object-oriented programming language type which provides XML-oriented data manipulation, wherein the object-oriented programming language type allows the combination of XML and object-oriented programming language type systems and is capable of accessing the XML data from within the object-oriented programming language by mapping the XML data to an object-oriented programming language object *(see Column 8: 33-37, "Thus the XML configuration information 116 in comment 114 in the example of lines 3 through 4 indicates that the public class Employee is to be mapped to the root element "Employee" in the XML grammar defined in employee.dtd."; Column 14: 47-55, "In step 340, a module is generated that marshals a data object of the class to one or more data items in the exchange format. For example, a JAVA module [object-oriented programming language type] is generated that marshals a data object of a JAVA class into one or more items of the XML configured by the DTD [XML-oriented data manipulation]. As a more specific example, a JAVA module is generated that exports an object of class Employee having attributes empname and empid into an XML element Employee having attribute empid and containing element LastName.")*; and

- an XML transformation capable of transformation a source type to a target type, wherein the source and target types can be either an XML type or an object-oriented programming language type *(see Column 9: 26-31, "One or more methods of the user-defined doclet 124 produces statements 130 for an XML DTD or Schema document based on the XML*

configuration data 116 in the comment statement 114 for a data object in the modified source code file 112 and on the neighboring JAVA class definition statement.").

As per **Claim 16**, the rejection of **Claim 15** is incorporated; and Hershberg further discloses:

- a global registry of XML transformations capable of looking up an existing XML transformation between a source and a target type *(see Column 11: 16-19, "In some of these embodiments, the conventional approach used employs not only the DTD file but also a mapping file that maps DTD elements/attributes to JAVA class attributes.")*.

As per **Claim 17**, the rejection of **Claim 15** is incorporated; and Hershberg further discloses:

- a library of XML transformations capable of looking up an existing XML transformation by name between a source and a target type *(see Column 11: 16-19, "In some of these embodiments, the conventional approach used employs not only the DTD file but also a mapping file that maps DTD elements/attributes to JAVA class attributes.")*.

As per **Claim 18**, Hershberg discloses:

- an XML data *(see Figure 1A: 118; Column 7: 54-58, "The XML configuration information 116 includes the user-defined JavaDoc tag "@xml-root-element" that is an example of tag 118 ...")*;

- an XML schema which defines the XML data *(see Column 7: 54-58, "The XML configuration information 116 includes the user-defined JavaDoc tag "@xml-root-element" that is an example of tag 118, and the property "dtdFile." The property is set to a value "employee.dtd" given between the quotation marks in line 3.5.")*;

- a lightweight XML store capable of retaining the XML data as a searchable index *(see Column 1: 54-62, "Data exchange also occurs between independently developed applications running on different families of processors and a database server. Such data exchange is often accomplished with a published (open) database query language. For example, the Structured Query Language (SQL) is used for exchanging data with database servers of several database systems. As used herein, a data exchange format includes either a markup language or a database query language or both.")*; and

- an object-oriented programming language type which corresponds to the XML schema and provides XML-oriented data manipulation, wherein the object-oriented programming language type allows the combination of XML and object-oriented programming language type systems and is capable of referencing the lightweight XML store and accessing elements of the XML data from within the object-oriented programming language *(see Column 14: 47-55, "In step 340, a module is generated that marshals a data object of the class to one or more data items in the exchange format. For example, a JAVA module [object-oriented programming language type] is generated that marshals a data object of a JAVA class into one or more items of the XML configured by the DTD [XML-oriented data manipulation]. As a more specific example, a JAVA module is generated that exports an object of class Employee having*

attributes empname and empid into an XML element Employee having attribute empid and containing element LastName.").

As per **Claim 19**, Hershberg discloses:

- an XML data *(see Figure 1A: 118; Column 7: 54-58, "The XML configuration information 116 includes the user-defined JavaDoc tag "@xml-root-element" that is an example of tag 118 ...")*;

- an XML schema which defines the XML data *(see Column 7: 54-58, "The XML configuration information 116 includes the user-defined JavaDoc tag "@xml-root-element" that is an example of tag 118, and the property "dtdFile." The property is set to a value "employee.dtd" given between the quotation marks in line 3.5.")*;

- a lightweight XML store capable of retaining the XML data at the text or tag level *(see Column 1: 54-62, "Data exchange also occurs between independently developed applications running on different families of processors and a database server. Such data exchange is often accomplished with a published (open) database query language. For example, the Structured Query Language (SQL) is used for exchanging data with database servers of several database systems. As used herein, a data exchange format includes either a markup language or a database query language or both.")*; and

- an object-oriented programming language type which corresponds to the XML schema and provides XML-oriented data manipulation, wherein the object-oriented programming language type allows the combination of XML and object-oriented programming language type systems and is capable of referencing the lightweight XML store and accessing

elements of the XML data from within the object-oriented programming language *(see Column 14: 47-55, "In step 340, a module is generated that marshals a data object of the class to one or more data items in the exchange format. For example, a JAVA module [object-oriented programming language type] is generated that marshals a data object of a JAVA class into one or more items of the XML configured by the DTD [XML-oriented data manipulation]. As a more specific example, a JAVA module is generated that exports an object of class Employee having attributes empname and empid into an XML element Employee having attribute empid and containing element LastName.").*

As per **Claim 20**, the rejection of **Claim 19** is incorporated; and Hershberg further discloses:

- the lightweight XML store is capable of representing the retained XML data as a hierarchical structure *(see Column 13: 39-61).*

As per **Claim 21**, the rejection of **Claim 20** is incorporated; and Hershberg further discloses:

- the hierarchical structure can be a tree *(see Column 13: 39-61).*

As per **Claim 22**, the rejection of **Claim 19** is incorporated; and Hershberg further discloses:

- the object-oriented programming language type is capable of accessing the XML data incrementally *(see Column 14: 61-65, "In step 350, a module is generated that de-marshals a*

data object of the class from one or more data items in the exchange format. For example, a JAVA module is generated that de-marshals a data object of a JAVA class from one or more data items of the XML configured by the DTD.").

Claims 23 and 25-36 are method claims corresponding to the computer-implemented system claims above (Claims 1 and 3-14) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 1 and 3-14.

Claims 37-39 are method claims corresponding to the computer-implemented system claims above (Claims 15-17) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 15-17.

Claim 40 is a computer-implemented method claim corresponding to the computer-implemented system claim above (Claim 18) and, therefore, is rejected for the same reason set forth in the rejection of Claim 18.

Claims 41-43 are computer-implemented method claims corresponding to the computer-implemented system claims above (Claims 19, 21, and 22) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 19, 21, and 22.

Claims 44 and 46-57 are machine readable storage medium claims corresponding to the computer-implemented system claims above (Claims 1 and 3-14) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 1 and 3-14.

Claims 58-60 are machine readable storage medium claims corresponding to the computer-implemented system claims above (Claims 15-17) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 15-17.

Claim 61 is a machine readable storage medium claim corresponding to the computer-implemented system claim above (Claim 18) and, therefore, is rejected for the same reason set forth in the rejection of Claim 18.

Claims 62-64 are machine readable storage medium claims corresponding to the computer-implemented system claims above (Claims 19, 21, and 22) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 19, 21, and 22.

Claim Rejections - 35 USC § 103

25. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

26. **Claims 2, 24, and 45** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Hershberg** in view of **US 6,754,884 (hereinafter "Lucas")**.

As per **Claim 2**, the rejection of **Claim 1** is incorporated; however, Hershberg does not disclose:

- the compiler is capable of generating the object-oriented programming language type based on the definition of a web service method.

Lucas discloses:

- the compiler is capable of generating the object-oriented programming language type based on the definition of a web service method *(see Column 2: 46-51, "Accordingly, the present invention provides a mechanism for manipulating both XML and native language objects, that is particularly well suited for the increasingly ubiquitous problem of mapping XML objects into and out of software applications and web services written in modern programming languages."; Column 6: 19-23, "Such a language mapping may be desirable in situations where, for example, a system having an internal operating environment based upon a programming language such as Java, is required to exchange data with other systems using a data representation language such as XML.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Lucas into the teaching of Hershberg to include the compiler is capable of generating the object-oriented programming language type based on the definition of a web service method. The modification would be obvious because one of ordinary skill in the art would be motivated to manipulate XML within the context of computer program applications and web services *(see Lucas – Column 1: 13-23).*

Claims 24 and 45 are rejected for the same reason set forth in the rejection of Claim 2.

Response to Arguments

27. Applicant's arguments filed on February 9, 2009 have been fully considered, but they are not persuasive.

In the Remarks, Applicant argues:

a) Applicant respectfully submits that the user-defined doclet that is included in the JavaDoc process, as disclosed in Hershberg (Figures 1-2 and Column 9, Lines 21-32 and 56-57, and Table 3), does not implement an object-oriented programming language type (Such as a JAVA type) that provides XML-oriented data manipulation. As disclosed in Paragraph [0027] and [0032], an object-oriented programming language type such as an XMLObject JAVA type allows accessing and manipulating the XML data from within Java using the combination of XML and Java type systems.

Hence, the object-oriented programming language type in the present invention is not a document type declaration in a DTD file for an XML processor or a user-defined doclet as disclosed in Hershberg. Therefore, Hershberg can not anticipate or render the present invention obvious, and independent claim 1 should be in allowable condition.

Examiner's response:

a) Examiner disagrees. Applicant's arguments are not persuasive for at least the following reasons:

First, with respect to the Applicant's assertion that Hershberg cannot anticipate the present invention, the Examiner respectfully submits that Hershberg clearly discloses an object-oriented programming language type *(see Column 14: 47-55, "In step 340, a module is generated that marshals a data object of the class to one or more data items in the exchange format. For example, a JAVA module is generated that marshals a data object of a JAVA class into one or more items of the XML configured by the DTD. As a more specific example, a JAVA*

module is generated that exports an object of class Employee having attributes empname and empid into an XML element Employee having attribute empid and containing element LastName."). Note that the JAVA module (object-oriented programming language type) marshals a data object of a JAVA class into one or more items of the XML configured by the DTD (XML-oriented data manipulation).

Second, the claims recite only "object-oriented programming language type" with no further clarification on the claim scope of the term "type" as intended by the Applicant to cover. Thus, as the claims are interpreted as broadly as their terms reasonably allow (see MPEP § 2111.01(I)), the interpretation of a broad limitation of an "object-oriented programming language type" as a JAVA module and the like by one of ordinary skill in the art is considered to be reasonable by its plain meaning.

Third, Applicant is attempting to import limitations from the specification by referring to the specification. However, according to MPEP § 2111.01(II), it is improper to import claim limitations from the specification that are not part of the claims.

Therefore, for at least the reasons set forth above, the rejections made under 35 U.S.C. § 102(e) with respect to Claims 1, 15, 18, 19, 23, 37, 40, 41, 44, 58, 61, and 62 are proper and therefore, maintained.

In the Remarks, Applicant argues:

b) On the other hand, as cited by the Examiner, Column 10, Lines 12-15 of Hershberg describes an example XML document with statements using a DTD file. Applicant respectfully submits that a statement in an XML document, as in Hershberg, is not an object-oriented

programming language type, such as a JAVA type. Therefore, Hershberg can not anticipate the present invention as embodied in Claims 2-3 and 13. Hence, Claims 2-3 and 13 should be in allowable condition.

Examiner's response:

b) Examiner disagrees. With respect to the Applicant's assertion that Hershberg cannot anticipate the present invention as embodied in Claims 3 and 13, the Examiner respectfully submits that the Examiner has addressed the Applicant's argument in the Examiner's response (a) hereinabove. Applicant's argument with respect to Claim 2 has been considered but is moot in view of the new ground of rejection.

Therefore, for at least the reason set forth above, the rejections made under 35 U.S.C. § 102(e) with respect to Claims 3 and 13 are proper and therefore, maintained.

Conclusion

28. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM. The Examiner can also be reached on alternate Fridays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Q. C./

Examiner, Art Unit 2191

/Wei Y Zhen/

Supervisory Patent Examiner, Art Unit 2191